САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Кафедра технологии программирования

**Воробьев Олег Александрович**

**Выпускная квалификационная работа бакалавра**

# Автоматизация управления системой тестирования знаний в многофилиальной организации

Направление 010300
Фундаментальная информатика и информационные технологии

Научный руководитель,
старший преподаватель
Севрюков С.Ю.

Санкт-Петербург
2016

# Содержание

# Введение

В настоящее время, в связи с постоянным возрастанием сложности вычислительных задач и необходимостью работы с данными больших объемов, все большее распространение получают технологии распределенной обработки данных. В условиях неконтролируемого прироста информации и постоянного увеличения необходимых для ее обработки ресурсов возникает потребность в непрерывном горизонтальном масштабировании распределенных систем. В связи с этим обнаруживается ряд требований к подобным системам, например: необходимость быстрого введения в эксплуатацию новых элементов распределенной системы, возможность изменения ее архитектуры без ущерба текущей работоспособности и автоматизация вышеприведенных процессов.

Данная работа посвящена изучению современных средств Управления конфигурацией IT-инфраструктуры и выполнению задачи по внедрению потенциально наиболее подходящей из них в разработанную ранее распределенную систему. В рамках работы будет сделан обзор компонентов внутренней архитектуры рассматриваемого приложения, его инфраструктуры и действующих в настоящий момент способов управления ею. Будут сформулированы задача и конечная цель относительно автоматизации управления данной системой, на основании которых будет произведено изучение доступных на рынке средств Управления конфигурацией и выбраны наиболее перспективные из них в условиях поставленной цели. Выбранными средствами будет осуществлена автоматизация текущего процесса управления системой с последующими выводами о пригодности данных средств при решении подобных задач. Текст работы разделен на главы и обладает следующей структурой:

• Глава 1 посвящена всестороннему обзору рассматриваемого приложения и действующим процессам управления его инфраструктурой. Целью главы является формулирование
задачи по модернизации данных процессов и требований к средствам, с помощью которых модернизация будет производиться.

• В Главе 2 производится знакомство читателя с понятиями DevOps и Управление конфигурацией (Configuration management), рассматриваются существующие средства Управления конфигурации, и, на основании выводов Главы 1, выбираются наиболее перспективные из них для достижения поставленных целей.

• В Главе 3 описывается весь процесс применения выбранных средств Управления конфигурацией, подытоживаются решенные с их помощью задачи и делается вывод о дальнейшей перспективе использования подобных средств.

• Глава 4 посвящена модернизации разработанных сценариев по применению средств Управления в условиях необходимости автоматического изменения управляемой конфигурации.

# Постановка задачи

Задачей данной работы является автоматизация управления инфраструктурой уже разработанного приложения с распределенной архитектурой: Системы тестирования знаний по русскому языку СТКПлюс.

Конечная цель работы: снижение трудозатрат и потенциального количества ошибок в процессах развертывания, модернизации и поддержки работоспособности инфраструктуры системы. Необходимо максимально автоматизировать процессы дистрибуции, обновления и функционирования данного приложения в сети независимых филиалов, с учетом его особенностей. Данную задачу предлагается разделить на следующие этапы:

1. Изучение внутренней архитектуры Системы тестирования с целью формирования требований к средству Управления конфигурацией:
    - Изучение внутренней архитектуры приложения;
    - Изучение инфраструктуры приложения;
    - Изучение действующих принципов управления данной инфраструктурой;
    - Анализ поставленной задачи в рамках полученной информации;
    - Формирование требований к средству Управления конфигурацией на основе проведенного анализа;

2. Обзор существующих средств Управления конфигурацией с целью выбора наиболее перспективного, с точки зрения поставленных целей:
    - Введение в принципы Управления конфигурацией;
    - Обзор представленных на рынке средств;
    - Анализ перспективности использования каждого из них в соответствии с сформированными требованиями;

• Обоснованный выбор одного или нескольких средств Управления конфигурацией для дальнейшего изучения;

3. Практическое изучение выбранных средств автоматизации:
   • Развертывание каждого из выбранных средств Управления конфигурацией на тестовом сетевом окружении с учетом архитектуры приложения;
   • Тестирование возможности выполнения всех элементов поставленной задачи данными средствами;
   • Проверка работоспособности данных средств для различных условий среды;
   • Вывод о достигнутых целях и дальнейшей перспективе использования данных средств;

# Глава 1. Постановка требований к системе управления конфигурацией с учетом текущей архитектуры ПО

## 1.1 Изучение текущей внутренней архитектуры и инфраструктуры приложения

Рассматриваемая система представляет собой программный комплекс для проведения тестирования, проверки, анализа полученных результатов и подготовки специализированных печатных форм, выполненный в виде многомодульной клиент-серверной информационной системы.

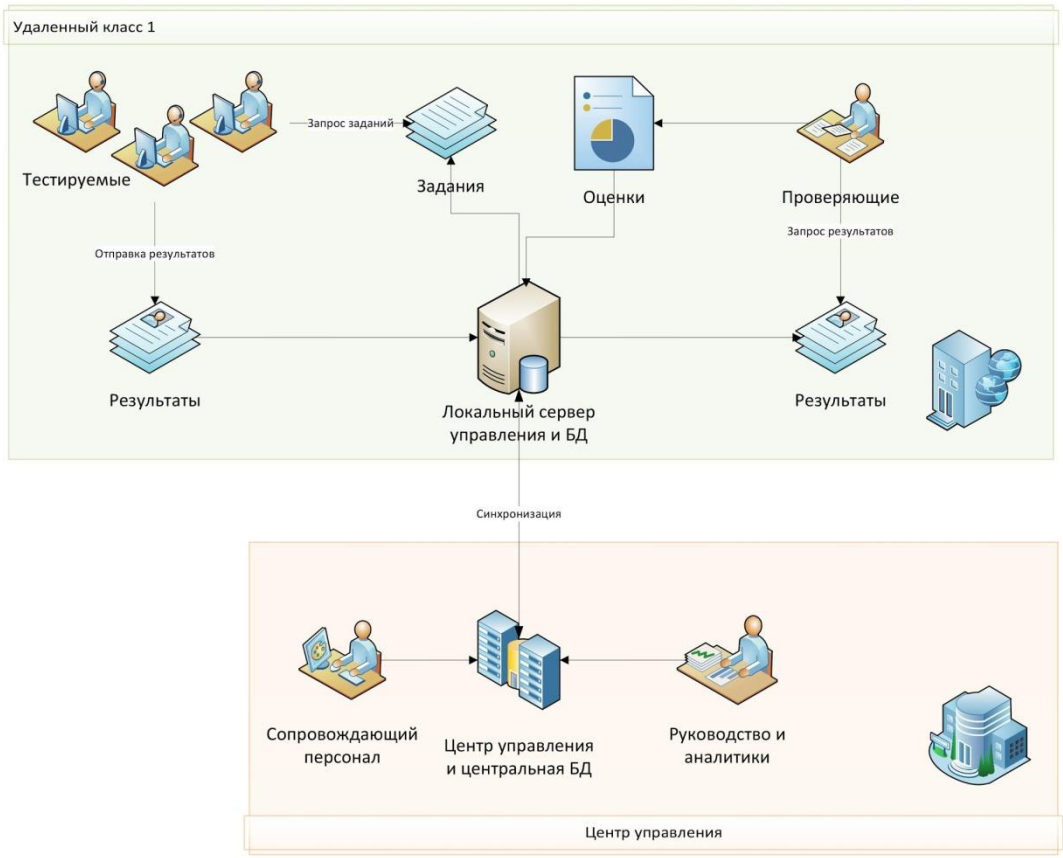


Рис. 1: Распределенная архитектура системы

Приложение обладает распределенной архитектурой с наличием единого центрального и множества локальных, взаимодействующих с ним

серверов, расположенных на удаленных филиалах. Помимо локального сервера, каждый удаленный филиал подразумевает наличие узлов с одним из двух установленных компонентов Системы: “Рабочее место тестируемого”, “Рабочее место преподавателя”, подключенных к хранилищу данных сервера филиала (см. Рис. 1).

В качестве хранилища данных, на каждом из локальных серверов используется СУБД MS SQL Server. Компоненты системы разработаны на платформе .NET на языке C#. Пользовательский интерфейс приложения представлен в виде однооконного настольного Windows приложения (модуль тестирования) и многооконного настольного Windows приложения (модуль управления, анализа и печати).

Основным функционалом системы является:

- Подготовка и развёртывание класса для проведения тестирования с использованием компьютеров под управлением Windows с современными мультимедийными возможностями, объединённых в локальную вычислительную сеть.
- Регистрация тестируемых и дистанционное управление сессиями тестирования на компьютерах тестируемых.
- Формирование тестов (вопросы, варианты ответов и т.д.).
- Тестирование с использованием вопросов различного типа и интерактивных материалов (изображения, аудио и видео-контент).
- Автоматическая и ручная проверка результатов.
- Формирование отчётных документов произвольной формы (встроенный конструктор печатных форм и отчётов).
- Анализ накапливаемых данных (сводные таблицы, диаграммы, контекстная разметка).

## 1.2 Изучение действующих принципов управления инфраструктурой приложения

В настоящий момент процесс развертывания локального сервера, компонентов системы, последующая поддержка работоспособности и модернизация происходят полностью в ручном режиме. Развертывание инфраструктуры отдельно взятого филиала производится в несколько этапов:

Первый этап, настройка сервера баз данных:

- Получение удаленного доступа к серверу через приложение TeamViewer с помощью сотрудников филиала
- Установка SQL Server 2014 Express WT.
- Настройка разрешающих правил брандмауэра порту 1433.
- Включение и настройка протокола TCP/IP
- Развертывание базы данных филиала и необходимых файлов
- Создание сетевого ресурса в рамках филиала с помощью “общей папки” и сетевого диска

Второй этап, подключение сетевого диска и настройка программных компонентов системы на узлах филиала: Развертывание необходимого ПО производится с помощью средства ClickOnce, позволяющего проводить установку Windows-приложений в автоматическом режиме после ручного запуска единственного скрипта. Подобные скрипты написаны для каждого из компонентов Системы и запускаются вручную на каждом из необходимых узлов сотрудниками филиала.

Третий этап, тестирование: После того, как проведена настройка локального сервера, и установлены необходимые компоненты на клиентские машины, сотрудниками филиала осуществляется попытка подсоединения к

локальному серверу с одного из узлов. В случае успеха, процесс развертывания филиала считается завершенным.

Мониторинг конфигураций филиалов также осуществляется в ручном режиме: сведения о текущих конфигурациях хранятся в сетевом документе сервиса GoogleDocs и обновляются администратором вручную. Коммуникация между администратором и сотрудниками филиалов, необходимая для координации процесса развертывания осуществляется при помощи электронной почты.

## 1.3 Анализ поставленной задачи

В документации системы сформулированы четкие требования для ручного развертывания отдельного филиала:

- Хост под управлением ОС Windows с возможностью удалённого подключения по RDP или TeamViewer.
- Установленный MS SQL Server Express edition или, в случае отсутствия, доступ и права на установку удалённо.
- Хост под управлением ОС Windows с возможностью удалённого подключения, конфигурация которого аналогична или приближена к той, что используется в компьютерных классах для тестирования в данный момент.
- Организация сетевого файлового ресурса доступного на обеих машинах, а в будущем для всех компьютеров класса и компьютеров проверяющих.

Исходя из данных требований и архитектуры системы, следует подчеркнуть, что для каждого из ее компонентов требуется хост под управлением ОС Windows. Этот факт является важным, поскольку, как

можно будет увидеть в последствии, большинство средств Управления конфигурацией изначально рассчитано на управление инфраструктурой, компоненты которой находятся под управлением UNIX-систем, наиболее часто используемых в серверной архитектуре. Работа с серверами, функционирующими под управлением ОС Windows, у большинства средств Управления оказывается нетривиальной в сравнении с UNIX. При этом, в рамках текущей задачи, очевидно, требуется всесторонняя поддержка ОС Windows для автоматизации дистрибуции и управления Системой.

На основании рассмотренного процесса ручного развертывания системы необходимо отметить, что, на данный момент, наиболее трудозатратным и незащищенным от возможных ошибок является процесс конфигурирования локального сервера филиала. Таким образом, в рамках текущей архитектуры Системы тестирования, задача автоматизации управления системой главным образом сводится к задаче автоматизации дистрибуции и управления локальными серверами филиалов с помощью средств Управления конфигурацией.

Также исходя из вышеозначенных требований к ручной установке и поставленной задачи автоматизации следует подчеркнуть необходимость минимизации ручного взаимодействия с локальными серверами системы при помощи RDP, TeamViewer и любых других инструментов удаленного доступа. В идеальном случае, для развертывания локального сервера с применением средств Управления конфигурацией, на стороне локального сервера не должно происходить ручного взаимодействия, кроме установки ОС Windows.

## 1.4 Итоговые требования к Средствам управления конфигурацией

Учитывая внутреннюю архитектуру компонентов, распределенную архитектуру приложения и описанные выше требования для ручной дистрибуции, в рамках поставленной задачи можно выделить следующие основные требования к ПО Управления конфигурацией:

1. Возможность работы с файловой системой;
2. Возможность изменения сетевого окружения;
3. Возможность автоматической дистрибуции, конфигурирования и запуска приложений;
4. Возможность выполнения вышеозначенных задач на узле под управлением ОС Windows;
3. Минимизация необходимости ручной работы непосредственно на развертываемом узле;

# Глава 2. DevOps.Управление конфигурацией. Выбор средств

## 2.1 Методология DevOps и Управление конфигурацией

Для решения задач введения в эксплуатацию отдельных элементов распределенных систем, а также возможности изменения архитектуры распределенных приложений вне зависимости от этапа их жизненного цикла в рамках серии встреч «DevOps Days» в 2009 году была предложена новая методология разработки Программного Обеспечения DevOps [1] (development and operations: разработка и IT-операции), основанная на идеи тесной взаимосвязи разработки и эксплуатации программного обеспечения.

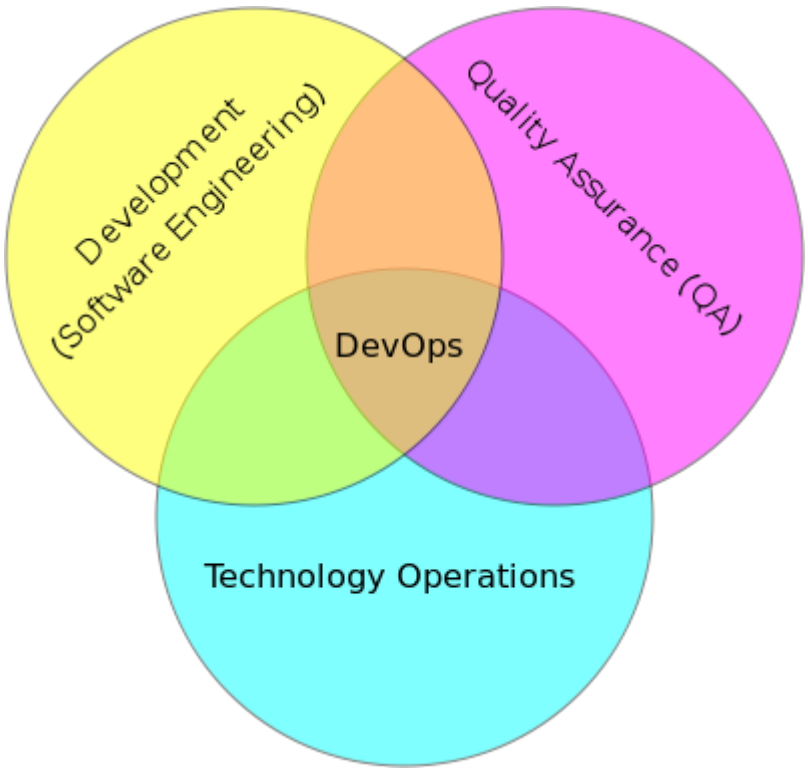


Рис. 2: Принцип методологии DevOps (взято с http://devopswiki.net)

Главной целью DevOps является создание единого цикла взаимозависимости между разработкой, развертыванием и эксплуатацией ПО для помощи организациям в быстром создании и обновлении их программных продуктов, используемых в режиме реального времени. Данная цель достигается путем создания между отделом разработки, IT-отделом и заказчиком «промежуточного звена» (см. Рис. 1), ответственного за оптимизацию всех этапов производства приложения, нуждающегося в непрерывном обновлении.

Существует несколько различных моделей внедрения **DevOps** в цикл разработки, развертывания и поставки Программного Обеспечения [2]. Все предложенные модели подразумевают тесную интеграцию IT-отдела с Отделом разработки: ведение общей хронологии вносимых в продукт изменений, взаимные тренинги, первоочередное ориентирование на возможности IT-отдела и так далее, вплоть до объединения двух команд. При этом, в каждой из предложенных моделей прослеживается необходимость непрерывно контролировать инфраструктуру создаваемого ПО, возможность ее модернизации в кратчайшие сроки в соответствии с постоянно изменяющимися требованиями.

Для решения данной задачи используется комплекс методов под общим названием «Управление Конфигурациями» («Configuration management») [3]. Конечной целью данного комплекса является представление инфраструктуры ПО частью создаваемого приложения, с возможностью автоматического контроля над всеми изменениями, происходящими в ней. В этом случае, инфраструктура приложения описывается в виде программного кода и находится под управлением Системы контроля версий. Представление инфраструктуры ПО в том виде, в котором с ней может свободно взаимодействовать разработчик, позволяет создать то самое необходимое «промежуточное звено» между IT-отделом и

Отделом разработки. При подобном подходе появляется возможность автоматизации изменений в инфраструктуре приложения в зависимости от изменений в его архитектуре, что позволяет решать проблемы, свойственные постоянно модернизирующимся распределенным системам.

В соответствии с сформулированными в предыдущей главе требованиями, необходимо проанализировать представленные на рынке средства Управления конфигурации с целью выбора наиболее подходящих из них в рамках поставленной задачи. В данной главе будут рассмотрены уже зарекомендовавшие себя к настоящему времени решения, осуществлен их сравнительный анализ и выбраны наиболее подходящие из них для дальнейшего, практического изучения.

## 2.2 Общие группы свойств рассматриваемых инструментов

Перед тем, как перейти к обзору конкретных средств Управления конфигурацией, следует выделить основные группы свойств существующих решений, на основании которых, в дальнейшем, можно будет проследить отличия между представленными средствами.

### Свойства описания желаемой конфигурации

В качестве данных свойств следует рассматривать свойства методов, с помощью которых осуществляется ввод данных о желаемом состоянии управляемых серверов, либо о необходимых изменениях в существующей конфигурации. Предлагается выделить следующие свойства:

1. Стиль описания управляемой конфигурации (декларативный или императивный);
2. Базовый язык, используемый для описания;

3. Наличие интерфейса, для визуального представления и управления текущей конфигурацией (Web-интерфейса);

**Свойства внедрения конфигурации в инфраструктуру**

Следующая группа свойств демонстрирует возможности средств Управления по внедрению уже описанных конфигураций в инфраструктуру.

1. Архитектура системы управления конфигурациями:
   - Централизованная: один управляющий узел;
   - Слабо распределенная: несколько управляющих узлов на множество управляемых;
   - Сильно распределенная: управляющий узел для каждого управляемого узла;

2.Метод доставки конфигурации:
   - “Pull”: Запрос конфигурации исходит от управляемого узла;
   - “Push”: Команда конфигурации исходит от управляющего узла;
   - Поддерживаемые платформы (список возможных платформ как для управляющих, так и для управляемых узлов);

**Свойства, значимые в рамках решения текущей задачи**

Также предлагается выделить группу свойств, учитывающих сформулированные в Главе 1 требования к используемому решению. К данным свойствам следует отнести:

1. Степень сложности работы с узлами под управление ОС Windows (в случае поддержки данной платформы);

2. Необходимость работ на управляемом узле (необходимость вручную проводить установку отдельных элементов Средства на управляемом узле);

## 2.3 Обзор средств Управления конфигурацией

### CFEngine

CFEngine [4] является первым из появившихся на рынке средств Управления Конфигурацией. CFEngine 1 разработан и представлен в 1993 году в Университете Осло (Норвегия) с целью автоматизации управления серверов кафедры Теоретической физики. В 1998 году представлена вторая версия фреймворка, учитывающая требования к подобным решениям, озвученные за прошедшее время и появившиеся новые парадигмы автоматизации управления. На сегодняшний день последней версией программы является, представленная в 2008 году, CFEngine 3, главным нововведением которой является концепция “Теории обещаний” [5] (“Promise theory”), используемая в качестве языка описания.

Данная концепция, в рамках CFEngine, подразумевает декларативный подход к управлению инфраструктурой с отсутствием непосредственных инструкций к находящимся под управлением узлам. На каждом узле содержатся файлы так называемых “обещаний” - файлы политик, описывающие состояние узла, в котором он обязуется находиться. Описание состояния происходит на собственном декларативном языке.

CFEngine представляет собой сильно распределенную систему с отсутствием управляющего узла. Компоненты системы, при помощи которых может централизованно производиться доставка файлов политик на

управляемые узлы и мониторинг соответствия описанному состоянию необязателен для успешного функционирования и взаимодействия узлов друг с другом. Изменение текущих политик может осуществляться как “Pull”, так и “Push” методами. Различные компоненты CFEngine [6] позволяют производить автоматическое развертывание, конфигурирование, мониторинг и взаимодействие управляемых узлов. Большинство компонент независимо друг от друга и то, какие из них будут установлены на конкретный узел, зависит от его роли в текущей инфраструктуре.

CFEngine распространяется как по Open-source, так и по коммерческой лицензии [7]. В Enterprise-версии Системы присутствует web-интерфейс, упрощающий работу с узлами. Рассмотренное средство поддерживает семейство Unix-систем, а также Mac OS X и Windows.

**Puppet**

Puppet [8] - средство управления конфигурацией, изначально сориентированное на использование в промышленных целях. Первая версия системы представлена в 2005 году. На текущий момент представлена уже четвертая версия системы.

Архитектура Puppet построена на взаимодействии двух отдельных модулей: Puppet Master, устанавливаемый на управляющие узлы, и Puppet Agent, устанавливаемый на управляемые. Подобное решение подразумевает либо централизованную, либо слабо-распределенную архитектуру и обеспечивает высокую степень горизонтального масштабирования.

В Puppet используется декларативный стиль описания конфигураций на собственном предметно-ориентированном языке, основанном на синтаксисе Ruby . Описание конфигураций осуществляется в так называемых

“манифестах”, куда заносятся данные о требуемом состоянии конкретного узла или группы узлов. Возможно объединение нескольких манифестов и всего необходимого программного окружения в отдельные модули. Существует официальная библиотека готовых модулей под решение конкретных задач, написанная сообществом Puppet [9]. В качестве методов доставки конфигурации данное решение сочетает “Push” и “Pull” технологии.

Существуют Open-source и Enterprise версии Puppet [10], отличающиеся наличием дополнительных встроенных компонент управления инфраструктурой (в частности Web-интерфейса), средств мониторинга, дополнительными модулями, доступными в официальной библиотеке. Данная система поддерживает операционные системы Unix, MacOC X, Windows для управляемых узлов и системы семейства Unix для управляющих.

**Chef**

Chef [11] - средство управления конфигурацией, изначально представленное в 2009 году, и, к настоящему времени, реализованное уже в двенадцатой версии. Chef обладает схожей с Puppet концепцией построения инфраструктуры, при этом отличаясь как в методах описания конфигурации, так и в методах ее внедрения.

Архитектура Chef, по аналогии с Puppet основывается на двух основных компонентах: Chef Server и Chef Client, функционирующих на управляющем и управляемом узлах соответственно. В случае с Chef, к данным компонентам добавляется Chef Workstation, предназначенный для управления взаимодействием двух предыдущих компонент. Управление осуществляется с помощью инструмента Knife, устанавливаемого на Chef Workstation. Также возможно создание изолированного управляемого узла с

помощью решения Chef Solo. Важным отличием данной системы является возможность автоматической дистрибуции Chef Client на управляемые узлы.

Chef использует императивный стиль описания конфигурации средствами языка Ruby: команды выполняются в той последовательности, в которой они написаны разработчиком. Присутствуют декларативные элементы. Инструкции по приведению узла в необходимое состояние содержатся в, так называемых, Cookbook`ах, объединяющих в себе набор рецептов - инструкций и требуемые файлы. Существует официальная библиотека Cookbook`ов, написанная сообществом Chef [13]. В общем случае, данная система использует метод доставки конфигурации по запросу: “Pull”. Возможность дистрибуции конфигураций одновременно на большое количество серверов “Push” методом реализуется с помощью отдельного расширения Enterprise-версии Chef.

Chef распространяется в трех возможных версиях: On premises Chef, Hosted Chef, Open Source Chef. Отличия On permieses Chef от Open Source версии заключаются в наличии дополнительных модулей развертывания, мониторинга и аналитики для управляемой инфраструктуры. Hosted Chef решение, помимо всех преимуществ enterprise версии включает предоставляемые виртуальные мощности для развертывания серверной части решения. В качестве поддерживаемых платформ для управляющего компонента выступают операционные системы семейства Unix, для управляемых компонентов и Chef Workstation - Unix, MacOC X, Windows.

**Ansible**

Ansible [14] является наиболее новым из обозреваемых Средств управления конфигурацией: проект был представлен в 2012 году. Система позиционируется как средство для решения повседневных задач системного

администрирования в силу простоты освоения и быстроты внедрения желаемых конфигураций.

Ключевой особенностью архитектуры Ansible является отсутствие “агента” на управляемом узле. Требования к управляемому узлу ограничиваются установленной на нем необходимой версии Phyton и возможностью доступа к нему по ssh. Таким образом, единственным возможным методом доставки конфигурации для Ansible является “Push”, а единственная возможная Ansible архитектура - централизованная.

Ansible использует декларативный стиль описания конфигурации формата YAML. Описание желаемых конфигураций для управляемых узлов осуществляется с помощью инвентарного файла, так называемых “Playbook`ов”, файлов ролей и файлов модулей. С помощью данных инструментов поддерживаются списки управляемых узлов, информация о том, к какому состоянию каждый из них необходимо привести и средства, которыми будет достигнут результат. Существует достаточно обширная библиотека готовых модулей для решения различных задач [15].

Ansible распространяется в Open Source и Enterprise (Ansible Tower) версиях. Ключевым отличием Ansible Tower [16] является наличие пользовательского интерфейса, реализующего автоматический запуск дистрибуции конфигураций, просмотр инфраструктуры, графиков развертывания и прочие прикладные задачи. Начиная с версии 1.7 Ansible предоставляет возможность работы с ОС Windows с помощью PowerShell [17], но, стоит заметить, что, в связи с относительно недавним началом поддержки, количество готовых Windows-решений не столь велико.

**SaltStack**

Система управления конфигурациями SaltStack [18] разработана в 2011 году разработчиками пакета скоростного обмена сообщениями ZeroMQ [19]. На основании данного решения построен сетевой уровень SaltStack. Основная идея данного средства схожа с идеей Ansible: максимально интуитивное и быстрое управление конфигурациями. При этом, ключевым отличием от предыдущего решения является наличие клиента, позволяющего выполнять с помощью SaltStack более комплексные задачи.

SaltStack основывается на двух компонентах: Salt Master, устанавливаемым на управляющий узел, и Salt Minion, устанавливаемом на управляемые узлы. Система подразумевает как централизованную, так и слабо-распределенную архитектуру управления конфигурациями, с возможностью распределения нагрузки между управляющими узлами. SaltStack сочетает “Pull” и “Push” методы доставки конфигурации с помощью собственного асинхронного протокола.

Описание требуемой к развертыванию конфигураций, в случае с SaltStack может носить как декларативный, так и императивный характер с возможностью, как отдавать управляемым узлам непосредственные команды на языке Phyton, так и описывать их состояние в формате YAML с последующим запуском при помощи модуля State.

В качестве возможного ПО для Salt Master выступают Unix-системы, а для Salt Minion также Mac OS X и Windows. Существует официальный репозиторий готовых windows-решений по дистрибуции необходимого ПО [20], сделанный по аналогии с Advanced Packaging Tool [21] для Linux, однако количество готовых решений является ограниченным. Визуальный интерфейс присутствует в Enterprise версии системы и обладает стандартным для средств Управления конфигурацией функционалом.

## 2.8 Сравнение представленных средств

Опираясь на выделенные ранее группы свойств инструментов Управления конфигурацией, были структурированы сведения, полученные в процессе изучения систем (Таблица 1, 2, 3):

| | CFEngine 3 | Puppet | Chef | Ansible | SaltStack |
|---|---|---|---|---|---|
| Стиль описания конфигурации | Декларативный | Декларативный | Императивный | Декларативный | Императивный/ Декларативный |
| Базовый язык | Собственный | Собственный (на основе Ruby) | Ruby | YAML | YAML и Python |
| Наличие интерфейса | В платной версии | В платной версии | В платной версии | В платной версии | В платной версии |

Таблица 1: Свойства описания конфигурации

| | CFEngine 3 | Puppet | Chef | Ansible | SaltStack |
|---|---|---|---|---|---|
| Архитектура системы | Сильно распределенная | Централизованная, слабо распределенная | Централизованная, слабо распределенная | Централизованная | Централизованная, слабо распределенная |
| Метод доставки | Pull, Push | Pull, Push | Pull, Push | Push | Push,Pull |
| Поддерживаемые платформы | Unix, MacOC X, Windows | Unix, MacOC X, Windows | Unix, MacOC X, Windows | Unix, MacOC X, Windows | Unix, MacOC, Windows |

Таблица 2: Свойства внедрения конфигурации

| | CFEngine 3 | Puppet | Chef | Ansible | SaltStack |
|---|---|---|---|---|---|
| Работа с узлами ОС Windows | Поддержка всеми типами узлов, наличие документации [22] | Поддержка всеми типами узлов, наличие документации [23], готовые решения от сообщества [9] | Поддержка управляемыми узлами, отсутствие отдельной документации, использование библиотеки Chocolatey [24, 25], готовые решения от сообщества [13] | Поддерживает, Наличие документации | Поддержка управляемыми узлами, наличие документации [26], отдельный репозиторий [20] |
| Необходимость работ на управляемом узле | Ручная установка клиента | Ручная установка клиента | Автоматическая установка клиента | Клиент отсутсвует | Ручная установка клиента |

Таблица 3: Свойства, значимые в рамках текущей задачи

В приведенных таблицах можно увидеть, что изученные решения слабо отличаются в принципах и возможностях управления распределенной инфраструктурой. Принципиально отличающейся архитектурой обладают CFEngine 3 и Ansible, что, тем не менее, позволяет предполагать их в качестве инструментов решения текущей задачи.

При более внимательном рассмотрении можно выделить ряд особенностей изученных систем, таких как: высокая степень масштабируемости CFEngine за счет сильно распределенной архитектуры, удобство управления инфраструктурой со множеством различных конфигураций Puppet с помощью специфического языка описания управления узлов, наличие максимально большого количества готовых решений Chef, отсутствие необходимости дистрибуции дополнительного ПО на управляемый узел Ansible, параллельное исполнение процессов SaltStack средствами сетевого уровня приложения ZeroMQ.

Согласно поставленным в Главе 1 требованиям и общей архитектуре приложения можно утверждать, что для выполнения текущей задачи необходима система с централизованной архитектурой, поддержкой ОС Windows для управляемых узлов и, при этом, возможностью максимально дистанцироваться от взаимодействия с ними. С учетом данных требований, отмеченных особенностей инструментов и изученных сравнительных статей [27, 28, 29], для дальнейшего практического изучения в рамках текущей задачи был выбран Chef, как средство управления, имеющее автоматическое развертывание части своих компонент и обширную библиотеку пользовательских решений, а также Puppet, как наиболее зрелый продукт, среди представленных решений с централизованной архитектурой.

## 2.9 Инструменты Управления конфигурации в среде Windows

Помимо рассмотрения популярных средств Управления конфигурацией, учитывая архитектуру системы, логично рассмотреть специализированные решения по автоматизации дистрибуции и поддержки ПО для ОС Windows. В рамках данной работы предлагается рассмотреть Chocolatey - менеджер пакетов в среде Windows совместно с инструментом автоматизации менеждмента Chocolatey-пакетов: Boxstarter [30] , которые будут использованы в практической части работы, а также System Center [31] - решение Microsoft по автоматическому развертыванию, обновлению и управлению Windows-узлами.

### Chocolatey и Boxstarter

Chocolatey представляет собой менеджер пакетов для ОС Windows аналогичный менеджеру пакетов Advanced Packaging Tool, используемому в Linux-средах. Данное решение позволяет производить дистрибуцию и

обновление Windows-приложений из специализированного каталога с помощью устанавливаемого Chocolatey-клиента. Процесс установки и обновления уже установленных программ происходит с помощью консольных команд, аналогично использованию apt-get в Linux.

Boxstarter является расширением Chocolatey, целью которого является полная автоматизация установки Chocolatey-пакетов, включая необходимые в процессе установки перезагрузки и принятие пользовательских соглашений. Как следствие, Boxstarter реализует возможность удаленной установки Chocolatey-пакетов и полностью автоматическое управление обновлениями.

Следует заметить, что в официальной документации Chef предлагается использование вышеозначенных инструментов совместно с данным средством Управления конфигурацией в решении задач управления Windows-узлами [24], что подчеркивает уместность использования Chocolatey и Boxstarter в решении поставленной задачи.

**Microsoft System Center**

Microsoft System Center представляет собой пакет решений для управления физическими и виртуальными Windows-средами. Данный пакет включает следующие решения:

- System Center Configuration Manager - инструмент автоматического развертывания приложений, мониторинга корректной работы и удаленного администрирования;
- System Center Operations Manager - инструмент развернутого мониторинга множества управляемых Windows-сред;
- System Center Data Protection Manager - инструмент автоматического резервного копирования управляемых сред;

• System Center Service Manager - инструмент внедрения стандартов MOF и ITIL в управляемых средах;

• System Center Virtual Machine - инструмент конфигурирования и управления виртуальными средами;

• ManagerSystem Center Orchestrator -инструмент администрирования и упорядочения процессов в управляемых средах;

• System Center Endpoint Protection - инструмент обеспечения защиты сред под управлением Configuration Manager;

Можно заметить, что совокупность данных решений полностью обеспечивает необходимый инструментарий для решения поставленной задачи автоматизации развертывания и управления в среде Windows. Однако, стоит учитывать, что составляющие данного пакета функционируют на основе службы каталогов Active Directory [32], что подразумевает постоянное нахождение в единой сети всех элементов распределенной инфраструктуры, в том числе центрального сервера, что в рамках обсуждаемой задачи невозможно из-за топологических и архитектурных особенностей слабосвязанных сетей филиалов (их самостоятельности и независимости).

# Глава 3. Практическое применение выбранных средств Управления конфигурацией

## 3.1 Развертывание систем на тестовом окружении

Для решения поставленной задачи, в рамках прохождения производственной практики в УСИТ СПбГУ, была предоставлена тестовая среда, состоящая из четырех узлов. Два узла находятся под управлением ОС Ubuntu 14.04 и предназначены для развертывания серверных приложений Chef и Puppet. Оставшиеся два узла управляются ОС Windows 7 и выступают в качестве локальных серверов филиала - управляемых узлов.

**Развертывание Puppet**

Согласно Главе 2, Puppet не подразумевает автоматической дистрибуции Puppet agent на управляемый узел: установка обеих компонент системы происходила в ручном режиме. Необходимо отметить, что для корректной работы системы необходимо соответствие версий Puppet master и Puppet agent [33]. Развертывание компонент производилось с помощью [33, 34, 35]. Процесс развертывания Puppet подробно описан в Приложении 1.

**Развертывание Chef**

В отличие от Puppet, Chef-client устанавливается на управляемый узел автоматически при наличии, в случае ОС Windows, WinRM доступа к узлу. Возможность наличия в системе множества точек доступа к серверу, которыми являются Chef-workstation, подразумевает наличие дополнительного протокола аутентификации между двумя данными компонентами системы: каждая рабочая станция ассоциирована с пользователем и организацией на сервере, что делает процесс

первоначального развертывания компонент системы менее интуитивным, чем развертывание компонентов Puppet. Само развертывание производилось на основании [36, 37] и подробно описано в Приложении 2.

## 3.2 Автоматизация развертывания сервера филиала

На основании документации системы были выделены следующие этапы развертывания сервера филиала, нуждающиеся в автоматизации в рамках текущей задачи:

1.Установка SQL сервер, настройка окружения:

- Установка SQL Server 2014 Express WT;
- Настройка разрешающего правила в брандмауэре для порта 1433;
- Включение протокола TCP/IP в диспетчере конфигурации сервера, в качестве TCP-порта указание: 1433;
- Создание разрешающего правила брандмауэра для входящего трафика на 1433 для активной сети;

2.Настройка сетевого окружения филиала:

- Создание локального пользователя disk с паролем disk;
- Создание общей папки C:/share/ на чтение и запись для пользователя disk;
- Создание сетевого диска T:/ на созданную общую папку;

3.Развертывание базы данных и файловой системы:

- Запуск конфигурационного SQL скрипта: examinator.sql;
- Извлечение содержимого файлового архива diskt.zip в созданную общую папку;

Обозначенные этапы были поочередно автоматизированы средствами Управления конфигурациями Puppet и Chef.

**Автоматизация развертывания сервера при помощи Puppet**

Автоматическое конфигурирование управляемых Puppet узлов осуществляется средствами так называемых “манифестов”. Каждый манифест содержит в себе следующие элементы языка описания конфигурации:

- Ресурсы: являются базовым инструментом приведения отдельных компонентов узла в необходимое состояние. Существует большое количество ресурсов для взаимодействия с файловой системой, управления системными службами и прочее. Каждый примененный ресурс, в соответствии с декларативным способом описания конфигурации Puppet, указывает, в каком состоянии должен находится тот или иной элемент управляемого узла.
- Классы: реализуют уровень абстракции, позволяя объединять различные ресурсы в единые конструкции для выполнения комплексных задач. Поддерживают механизмы наследования и переопределения атрибутов используемых ресурсов.
- Узлы (nodes): определяют, к каким именно управляемым узлам будут применяться те или иные ресурсы и классы. Также поддерживают механизм наследования, тем самым предоставляя гибкость в управлении сетевым окружением.
- Ссылки на манифесты: позволяют использовать классы и узлы, расположенные в другом манифесте.

Помимо данных элементов в Puppet существует понятие “модулей”. Модуль представляет собой набор каталогов, содержащих файлы, классы,

манифесты и сторонние библиотеки с целью их структуризации и единого применения к управляемым узлам.

Для автоматизации развертывания был создан манифест, логически разделенный на четыре отдельных класса:

• Класс fileUp реализует доставку на управляемый узел необходимых для развертывания файлов: установочного образа SQL Server 2014 Express, SQL скрипта, а также файлового архива. Доставка происходит средствами ресурса File, предназначенного для работы с файловой системой. Сами файлы хранятся на сервере в специально созданном модуле, с возможностью доступа к ним.

• Класс sqlInst реализует установку экземпляра SQL Server 2014 Express, а также настройку окружения. Существует специализированный модуль Puppet [38], предназначенный для решения данной задачи, но для его использования необходима Enterprise версия программы. По этой причине, в данном классе для установки экземпляра сервера и настройки окружения используется ресурс Exec, реализующий выполнение внешних команд, в данном случае: команд cmd.exe на управляемом узле.

• Класс usDiskSh отвечает за создание локального пользователя disk, общей папки share, предоставления доступа к ней и подключения сетевого диска T:/. Для реализации данных возможностей используются ресурсы File и Exec.

• Класс zipFilScr производит установку экземпляра 7zip с помощью ресурса Package, с последующим разархивированием необходимого архива в папку share и запуска начального скрипта ресурсом Exec.

**Автоматизация развертывания сервера при помощи Chef**

Были применены следующие средства описания процесса автоматического развертывания Chef:

• Cookbook: общий способ хранения конфигурации, предназначенной для развертывания на управляемых узлах. Содержит в себе так называемые “рецепты”, файлы атрибутов, шаблонов и метаданных. Каждый cookbook представляет собой готовое решение некоторой задачи на управляемых узлах. Как правило, “Поваренные книги” свободно распространяются сообществом пользователей Chef, следствием чего является наличие готовых решений для распространенных задач Управления конфигурацией. Существует также официальный репозиторий пользовательских решений [15], с возможностью быстрой установки необходимого Cookbook`a.

• Рецепт: файлы Ruby, в которых, при помощи языка DSL [42], описывается необходимое состояние управляемого узла. Каждый cookbook может содержать различное количество рецептов, по умолчанию запускается рецепт default.rb, при этом одни рецепты могут запускаться из других. Описание состояния узла в рецепте происходит при помощи так называемых “Ресурсов”.

• Ресурс: назначение Chef-ресурсов аналогично назначению Puppet-ресурсов, описанному выше: ресурсы Chef являются базовыми инструментами приведения отдельных компонентов узла в необходимое состояние. Ресурсы представляют собой лаконичную абстракцию над Провайдерами - внутренними компонентами Chef, автоматически определяющими, какие действия нужно произвести при вызове Ресурса, в зависимости от конкретной ситуации данного вызова.

• Атрибуты: файлы атрибутов представляют собой файлы хранения данных, необходимых для работы конкретного Cookbook`a. Данные файлы содержат информацию, использующуюся в рецептах, а также

производят генерирование динамических данных при запуске “Поваренной книги” (например: ip-адрес управляемого узла).

• Шаблоны: шаблоны предоставляют возможность автоматического генерирования общих файлов Cookbook`а на основе его атрибутов.

• Метаданные: файл метаданных содержит в себе всю базовую информацию о конкретном Cookbook`е, включая имя, версию, создателя, список зависимостей.

Кроме понятия Cookbook и его составляющих, в Chef также присутствует возможность создания так называемой “Роли” - списка запуска нескольких Cookbook`ов для приведения узла в необходимое состояние в соответствии с его “Ролью”.

Для автоматизации развертывания средствами Chef был создан отдельный Cookbook “localServerInstall” с несколькими рецептами, автоматизирующими отдельные этапы развертывания сервера:

• Рецепт mssql_port_rule.rb производит установку Boxstarter на управляемый узел. Далее, в данном рецепте с помощью Boxstarter производится автоматическая загрузка и установка SQL Server 2014 Express и настраивание окружения с помощью специализированных ресурсов registry_key и windows_firewall_rule.

• Рецепт user_dir_disk_share.rb отвечает за создание локального пользователя disk, создание папки C:\\share и настройку общего доступа к ней, подключение сетевого диска. Данные действия производятся с помощью ресурсов user, group, directory, batch.

• Рецепт db_script_files.rb при помощи Boxstarter производит установку 7zip для дальнейшего разархивирования файлов в общую папку, а также запуск SQL-скрипта. В данном рецепте используются

следующие ресурсы: сторонний ресурс boxstarter, directory, cookbook_file, batch.

Для возможности использования Boxstarter в рецептах, из репозитория пользовательских решений на Chef-сервер был установлен одноименный cookbook, с последующим добавлением зависимости от него в metadata.rb. Все используемые в Cookbook`е файлы были предварительно загружены в директорию files данного Cookbook`a.

## 3.3 Тестирование

Процесс тестирования полученных решений заключался в проверке корректности развертывания тестовых серверов на узлах с разной степенью изначальной готовности: начиная от развертывания на полностью неподготовленные узлы, заканчивая узлами, дистрибуция на которые уже была произведена. В ходе данного тестирования были выявлены и решены следующие проблемы:

- При автоматической конфигурации отдельных компонентов с обращением непосредственно к cmd.exe (Ресурсы Exec и Batch для Puppet и Chef соответственно) необходимо производить дополнительную проверку на предмет повторного конфигурирования.
- Существует известная проблема [39], связанная с именами запакованных файлов, содержащих в названии кириллицу, возникающая при разархивировании данных файлов средствами языка Ruby. Решением данной проблемы является работа с zip архивами средствами 7zip, что подразумевает необходимость установки данного ПО на управляемый узел, которая была осуществлена средствами Управления конфигурациями.

• В процессе дистрибуции Chef-client на управляемый узел на данном узле создается пользователь, из под учетной записи которого происходит дальнейшее автоматическое развертывание. В процессе тестирования было обнаружено, что данному пользователю недостаточно прав для установки экземпляра SQL Server 2014 Express. Данная проблема была решена автоматической установкой SQL Server с помощью Boxstarter.

## 3.4 Выводы

Оба представленных средства успешно справляются с поставленной задачей: С их помощью были разработаны два одинаковых по функционалу прототипа, автоматизирующих наиболее трудоемкую часть развертывания филиала. В совокупности с уже автоматизированной средствами ClickOnce настройкой программных компонентов, при условии дальнейшей доработки прототипов и их тестировании на большем количестве конфигураций, развертывание филиала системы будет происходить средствами минимального человеческого участия и не будет требовать специализированных знаний со стороны сотрудников филиала.

Сравнивая методы, которыми решается поставленная задача в том и в другом случае, можно заметить, что наличие в свободном доступе большого числа пользовательских решений в случае с Chef позволяет, в сравнении с Puppet, значительно сократить количество выполняемых внешних команд в процессе развертывания. Также стоит подчеркнуть, что, благодаря декларативному стилю описания конфигураций, поддержки наследования и отношению к управляемой инфраструктуре как к программному коду, Puppet, в сравнении с Chef, предоставляет более широкие возможности по управлению обширной инфраструктурой со множеством ролей.

Дальнейшие перспективы использования рассмотренных средств следует рассматривать в зависимости от последующих задач. Chef, благодаря обширной библиотеки готовых решений, является перспективным средством автоматизации развертывания сложных систем с использованием нестандартного ПО. Puppet представляет собой перспективное решение по управлению, а также непрерывному мониторингу и обновлению большого количества узлов в связке со специализированными средствами.

# Глава 4. Модернизация решений

В рамках тестирования на предмет возможности модернизации полученных решений, были использованы наработки коллеги-однокурсника, Иванова Кирилла, в задачу которого входила разработка специализированных средств для синхронизации файловых данных между центральным сервером и серверами филиалов, а также в рамках сети одного филиала.

## 4.1 Обновленный сценарий развертывания филиала

В процессе решения задачи синхронизации данных, Ивановым Кириллом были разработаны прототипы модулей передачи файлов внутри филиала и синхронизации файлов с центром управления.

Модуль передачи файлов в рамках филиала оформлен в виде Windows-службы с применением библиотеки SignalR [40] от Microsoft. Данный модуль представляет собой SignalR-сервер, обрабатывающий запросы клиента, которым является модифицированный модуль тестирования. Передача файлов между сервером филиала и клиентскими приложениями происходит с помощью “общей папки” Windows.

Модуль передачи файлов между сервером филиала и центральным сервером управления также представляет собой службу, функционирующую на локальном сервере филиала. Синхронизация файлов между локальным сервером и сервером управления происходит в сеансовом режиме с помощью ,библиотеки Sync Framework [41], ответственной за синхронизацию каталогов.

В соответствии с реализованными прототипами, Ивановым Кириллом был предложен модифицированный сценарий развертывания филиала:

1. Настроить сервер базы данных на сервере филиала;

2. Установить и настроить модифицированный модуль управления;
   - В конфигурационном файле DataEditor.config необходимо указать строку для подключения к базе;
   - В конфигурационном файле App.Config необходимо указать адрес сервера SignalR модуля передачи файлов;

3. Установить модуль передачи файлов внутри филиала. В реестре в разделе настроек модуля указать адрес сервера SignalR и местоположение, куда необходимо сохранять аудио-ответы с компьютеров для тестирования;

4. На каждом компьютере для тестирования настроить общую папку в сети и доступ к ней;

5. Установить модуль тестирования на каждый компьютер для тестирования. В конфигурационном файле App.Config указать имя компьютера в сети, локальную папку для сохранения и адрес сервера SignalR модуля передачи файлов;

6. Создать подключение VPN на сервере филиала для связи с центром средствами Windows;

7. На сервер филиала установить модуль синхронизации файлов с центром управления. В реестре в разделе настроек модуля указать

параметры подключения VPN, указать местоположение аудио-ответов на сервере филиала и адрес общей папки в сети центра.

## 4.2 Автоматизация обновленного сценария

Анализируя обновленный сценарий развертывания филиала, можно выделить этапы, автоматизируемые в рамках сервера филиала, на основе решений Главы 3. К данным этапам относятся:

- Настройка сервера базы данных на сервере филиала (реализовано);
- Модифицирование конфигурационного файла DataEditor.config;
- Установка и запуск модулей синхронизации;
- Создание VPN подключения;
- Настройка модулей в реестре;

В рамках использования Chef и Puppet, были найдены необходимые средства по автоматизации данных шагов. Следует отметить, что задача создания VPN подключения сводится к задаче редактирования содержимого файла rasphone.pbk, расположенному на сервере филиала по адресу:

“C:\Users\USERNAME\AppData\Roaming\Microsoft\Network\Connections\Pbk”.

Оставшиеся шаги на данном этапе не поддаются автоматизации в следствии отсутствия доступа средств Управления конфигурацией к клиентским узлам филиала.

**Автоматизация средствами Puppet**

Модифицирование существующего сценария развертывания заключалось в добавлении к существующему сценарию новых классов, реализующих автоматизацию сформулированных этапов:

- Средствами ресурса File автоматизировано редактирование файлов DataEditor.config и rasphone.pbk;
- Средствами ресурса Exec произведена установка, заранее скопированных в файловый модуль разработанных прототипов;
- Средствами ресурса Exec произведена установка и запуск, заранее скопированных в файловый модуль, разработанных прототипов;
- При помощи свободно распространяемого модуля Puppetlabs-registry осуществлена настройка необходимых параметров реестра;

**Автоматизация средствами Chef**

Учитывая доступные в Chef средства описания конфигурации, логично автоматизировать новые шаги сценария в виде отдельного Cookbook`a, для объединения с существующим решением в рамках Роли, которой локальный сервер будет соответствовать в итоге. Был создан Cookbook, реализующий следующий функционал:

- Автоматическое изменение содержимого файлов DataEditor.config и rasphone.pbk при помощи ресурса File;
- Автоматическая установка прототипов средствами ресурса Batch;
- Запуск установленных сервисов с помощью ресурса Windows_service;
- Редактирование параметров реестра ресурсом Registry_key;

## 4.3 Выводы

В результате тестирования модифицированных сценариев получен положительный результат, однако к моменту написания текста работы не удалось провести тестирование в реальной среде.

Рассматриваемыми средствами Управления конфигурацией была продемонстрирована возможность модификации существующих сценариев без ущерба текущему функционалу с возможностью немедленного последующего развертывания с изменением текущей конфигурации узлов. Наличие дополнительной абстракции “Роль” в Chef, позволяющей логически разделять законченные этапы развертывания, в случае с Puppet, компенсируется наличием механизма наследования для управляемых узлов. Следует заметить, что, как и в случае с первоначальным развертыванием филиала, Chef, в сравнении с Puppet, продемонстрировал большее количество готовых решений по управлению инфраструктурой, позволяющее свести к минимуму выполнение внешних команд в процессе развертывания.

# Заключение

В рамках данной работы была изучена архитектура существующей распределенной Системы тестирования знаний по русскому языку и действующие методы управления ее инфраструктурой. Были сформулированы задачи и цели автоматизации процессов управления системой средствами Управления конфигурацией.

Было произведено изучение понятий DevOps и Управление конфигурацией, рассмотрены существующие программные решения по Управлению конфигурацией с последующем сравнением перспективности в рамках решаемой задачи. С помощью двух выбранных средств была автоматизирована часть процесса развертывания системы с последующим сравнением полученных результатов и выводами о дальнейшей перспективности.

Были произведены модификации текущих сценариев автоматизации, с учетом включения результатов сторонней работы по синхронизации файловых ресурсов системы. В процессе модификации были продемонстрированы гибкость функционала рассматриваемых систем и перспективы дальнейшего использования. Полученные в ходе работы результаты, при условии доработки и дополнительного тестирования, могут быть использованы для автоматизации текущих процессов управления системой.

# Список источников

1. DevOps. http://devopswiki.net/index.php/DevOps

2. Модели развертывания DevOps. http://www.osp.ru/os/2013/05/13035991/

3.Управление конфигурацией.
http://www.sei.cmu.edu/productlines/frame_report/config.man.htm

4. Документация CFEngine. https://docs.cfengine.com/lts/

5. Теория обещаний в CFEngine.
http://www.networkworld.com/article/2449562/sdn/promise-theory-mark-burgess-cfengine-sdn-cisco-aci-apic-opflex.html

6. Архитектура CFEngine. https://docs.cfengine.com/lts/guide-introduction.html

7.Лицензии CFEngine. https://cfengine.com/product/

8 Документация Puppet. https://docs.puppet.com/

9. Библиотека решений Puppet. https://forge.puppet.com/

10 Лицензии Puppet. https://puppet.com/enterprise-and-open-source

11 Документация Chef . https://docs.chef.io/

12 Сравнение лицензий Chef. https://www.upguard.com/articles/chef-open-source-vs.-hosted-chef-vs.-on-premises-private-chef

13 Библиотека пользовательских решений Chef . https://supermarket.chef.io/

14 Документация Ansible. http://docs.ansible.com/

15 Модули Ansible. http://docs.ansible.com/ansible/list_of_all_modules.html

16 Ansible tower . https://www.ansible.com/tower

17. Ansible на Windows . https://www.ansible.com/windows

18 Документация SaltStack . https://docs.saltstack.com/

19 ZeroMQ. http://zeromq.org/intro:read-the-manual

20 Библиотека решений SaltStack для Windows.
https://docs.saltstack.com/en/latest/topics/windows/windows-package-manager.html

21 Advanced Packaging Tool. http://help.ubuntu.ru/wiki/apt

22 CFEngine, руководство Windows.
https://auth.cfengine.com/archive/manuals/st-windows
23 Puppet, руководство Windows. https://docs.puppet.com/windows/
24 Chef, руководство Windows . https://www.chef.io/solutions/windows/
25 Менеджер пакетов Chocolatey. https://chocolatey.org/
26 SaltStack, руководство Windows.
https://docs.saltstack.com/en/latest/topics/installation/windows.html
27 Сравнение инструментов управления конфигурацией.
http://www.infoworld.com/article/2609482/data-center/data-center-review-puppet-vs-chef-vs-ansible-vs-salt.html
28 Сравнение Puppet и Chef. https://www.upguard.com/blog/puppet-vs-chef-battle-wages
29 Сравнение Puppet и CFengine. https://www.upguard.com/blog/puppet-vs-cfengine
30 Boxstarter. http://boxstarter.org/WhyBoxstarter
31 System center. https://www.microsoft.com/ru-ru/server-cloud/products/system-center-2012-r2/overview.aspx
32 Доменные службы active directory. https://technet.microsoft.com/ru-ru/windowsserver/dd448614.aspx
33 Установка сервера Puppet.
https://docs.puppet.com/puppetserver/2.2/install_from_packages.html#quick-start
34 О Puppet server.
https://docs.puppet.com/puppetserver/latest/services_master_puppetserver.html
35 Установка Puppet для Windows.
https://docs.puppet.com/pe/latest/install_windows.html
36 Установка сервера Chef . https://docs.chef.io/install_server.html#standalone
37 Установка Chef за 12 шагов.
https://www.digitalocean.com/community/tutorials/how-to-set-up-a-chef-12-configuration-management-system-on-ubuntu-14-04-servers
38 Модуль Puppet: Sql server. https://forge.puppet.com/puppetlabs/sqlserv

39 Ruby. Название файла не в ASCII кодировке.
https://github.com/rubyzip/rubyzip/wiki/Files-with-non-ascii-filenames
40 SignalR . http://www.asp.net/signalr
41 Microsoft Sync Framework.
https://msdn.microsoft.com/ru-ru/library/bb902854(v=sql.105).aspx

# Приложение 1. Развертывание Puppet

**Развертывание Puppet master:**

1. Установка программы из репозиториев Advanced Packaging Tool командой: “sudo apt-get install puppetmaster”;

2. Редактирование конфигурационного файла Puppet master, расположенного по адресу: “puppetlabs/etc/puppet/puppet.conf”.
Добавление в данный файл следующей конфигурации:

“[master]
certname = master_cert_name”,

Где master_cert_name - произвольное имя узла;

3. Запуск Puppet server командой: “service puppetserver start”.

**Развертывание Puppet agent:**

1. Загрузка и установка дистрибутива поддерживаемой версии;

2. Редактирование файла “%SystemRoot%\system32\drivers\etc\hosts”.
Добавление в данный файл следующей конфигурации:

“xxx.xxx.xxx.xxx  master_cert_name”,

Где “xxx.xxx.xxx.xxx” - сетевой адрес Puppet master, master_cert_name - имя управляющего узла.

3.Редактирование файла настроек puppet:
“C:\ProgramData\PuppetLabs\puppet\etc\puppet.conf”
Добавление в данный файл следующей конфигурации:

“[main]
certname = agent_cert_name
server = master_cert_name
environment = production
runinterval = 1h”

Где agent_cert_name - произвольное имя управляемого узла.

4. Запуск службы на управляемом хосте: “puppet agent --test”
5. Подтверждение запроса  на сервере: “puppet cert sign agent_cert_name”
6. Выполнение на клиенте следующих команд:

“puppet agent --enable
puppet agent --test”

## Приложение 2. Развертывание Chef

**Развертывание Chef-server:**

1.Выполнение команды: “hostname -f”.

Результатом команды должен быть адрес, по которому сервер может быть доступен. Если это не так, необходимо отредактировать файл:

sudo nano /etc/hosts

Содержимое файла должно быть примерно следующим:

127.0.1.1 текущий_адрес текущий_буквенный_адрес
127.0.0.1 localhost

Необходимо модифицировать верхнюю строку с учетом текущего адреса узла и сохранить изменения.

2.Загрузка и установка необходимой версии Chef-server.

Ссылка: https://downloads.chef.io/chef-server/ubuntu/#/

Скопировать адрес ссылки на сайте в соответствии с версией ОС, вернуться в домашний каталог, (команда: cd ~), выполнить команду:

wget скопированная_ссылка

Установить загруженный пакет командой:
sudo dpkg -i chef-server-core_*.deb,

где * - версия скачанного пакета.

3. Запуск Chef -server.

Выполнить команду: sudo chef-server-ctl reconfigure

4: Создание администратора и организации.

Чтобы создать пользователя, необходимо выполнить команду:

chef-server-ctl user-create ИМЯ_ПОЛЬЗОВАТЕЛЯ ИМЯ ФАМИЛИЯ EMAIL ПАРОЛЬ

Чтобы создать организацию, необходимо выполнить команду:

sudo chef-server-ctl org-create КРАТКОЕ_НАЗВАНИЕ "ПОЛНОЕ НАЗВАНИЕ" --association_user ИМЯ_СОЗДАННОГО_ПОЛЬЗОВАТЕЛЯ -ИМЯ_ФАЙЛА_КЛЮЧА.pem

После выполнения данных команд в домашнем каталоге будут созданы два файла, необходимые для подключения к серверу:

ИМЯ_ПОЛЬЗОВАТЕЛЯ.pem, ИМЯ_ФАЙЛА_КЛЮЧА.pem

5: Установка management console для работы через web-интерфейс (необязательно).

Зайти по адресу сервера, выполнить предложенные команды.

**Настройка Chef-workstation:**

6: Установка системы контроля версий.

Выполнить следующие команды:

```
sudo apt-get update
sudo apt-get install git
```

7: Клонировать репозиторий Chef.

Выполнить команды:

```
cd ~
git clone https://github.com/chef/chef-repo.git
```

8: Установка скачанного репозитория под контроль версий.

Установить имя и почтовый адрес, для фиксации коммитов:

```
git config --global user.name "Ваше имя"
git config --global user.email "почта@domain.com"
```

Указать для каталога хранения ключей игнорирования контроля версий:

```
echo ".chef" >> ~/chef-repo/.gitignore
```

Перейти в необходимый каталог:

```
cd ~/chef-repo
```

Выполнить команду:

```
git add .
```

Сохранить изменения:

```
git commit -m "Excluding the ./.chef directory from version control"
```

9: Установка Chef Development Kit

Скопировать ссылку на нужную версию по данному адресу:

https://downloads.chef.io/chef-dk/ubuntu/#/

Выполнить:

```
cd ~
wget скопированная_ссылка
```

Установить загруженный пакет:

```
sudo dpkg -i chefdk_*.deb
```

где * - версия пакета.

Проверить работоспособность:

```
chef verify
```

10: Загрузка ключей аутентификации

Созданные на шаге 4 ключи скачать и поместить в папку ~/chef-repo/.chef

11: Конфигурирование Knife

Открыть файл knife.rb:

```
nano ~/chef-repo/.chef/knife.rb
```

Заполнить файл:

```
current_dir = File.dirname(__FILE__)
log_level                :info
log_location             STDOUT
node_name                "Имя вашей рабочей станции"
client_key               "#{current_dir}/КЛЮЧ_ПОЛЬЗОВАТЕЛЯ.pem"
validation_client_name   "КЛЮЧ_ОРГАНИЗАЦИИ.pem"
validation_key           "#{current_dir}/КЛЮЧ_ОРГАНИЗАЦИИ.pem"
chef_server_url          "https://server_domain_or_IP/organizations/Имя_организации"
syntax_check_cache_path  "#{ENV['HOME']}/.chef/syntaxcache"
cookbook_path            ["#{current_dir}/../cookbooks"]
```

12: Проверка работоспособности

cd ~/chef-repo
knife client list

В случае ошибки ssl, выполнить:

knife ssl fetch

Проверить повторно:
knife client list

Результатом должно быть имя вашей рабочей станции.

**Дистрибуция Chef Client**

13: Дистрибуция ноды.

Выполнить:

knife bootstrap windows winrm IP_адрес_ноды -x Имя_пользователя -P Пароль -N 'Желаемое имя ноды'

Проверить наличие ноды в списке:

knife node list

## Приложение 3. Исходный код сценариев

1. https://github.com/nevitia/chef-script
2. https://github.com/nevitia/puppet-script